FAT-Pointer based range TLB

Akilan Selvacoumar[1], Robert Stewart[1], Hans-Wolfgang Loidl[1], and Ryad Soobhany[1]

[1]Heriot-Watt University, UK

Abstract

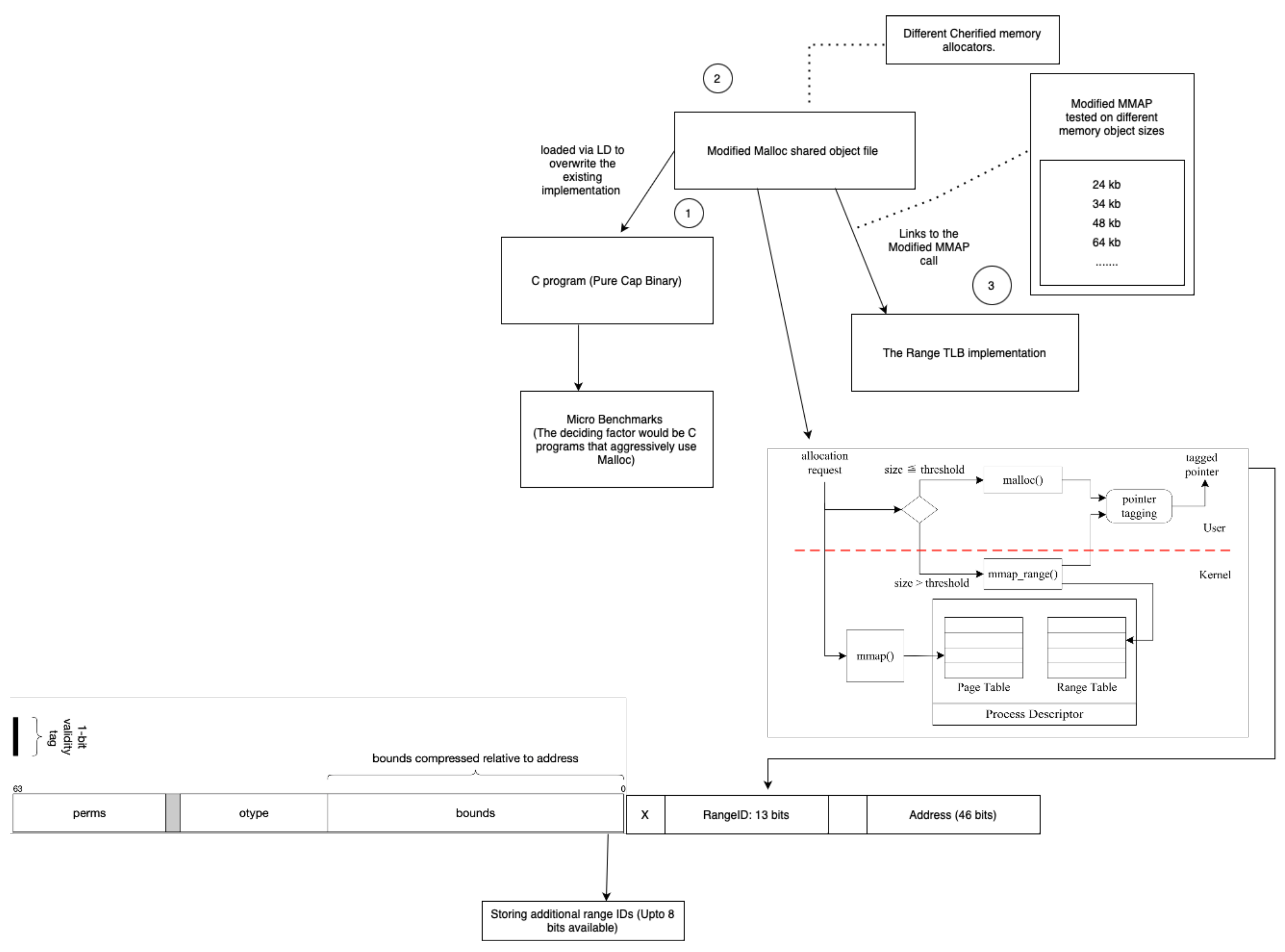
The CHERI architecture is designed to provide security guranetees, This has had performance implications such as more L1 TLB misses on purecap mode rather than running an application as a 64 bit program. The following expirement proposes on adding range TLB which is stored on the page table to store TLB translation entries of large memory objects (This would inturn lookup TLB entries in the range table rather than looking up the L1 TLB entry). During this process the range ID would be added to the pointer which allows looking up the TLB entries at an earlier stage during the pipeline. The FAT pointer structure of CHERI allows to encode custom bits of meta-data which in this case can be used to store the ID of the range TLB entry.

Research Questions

Research Questions:

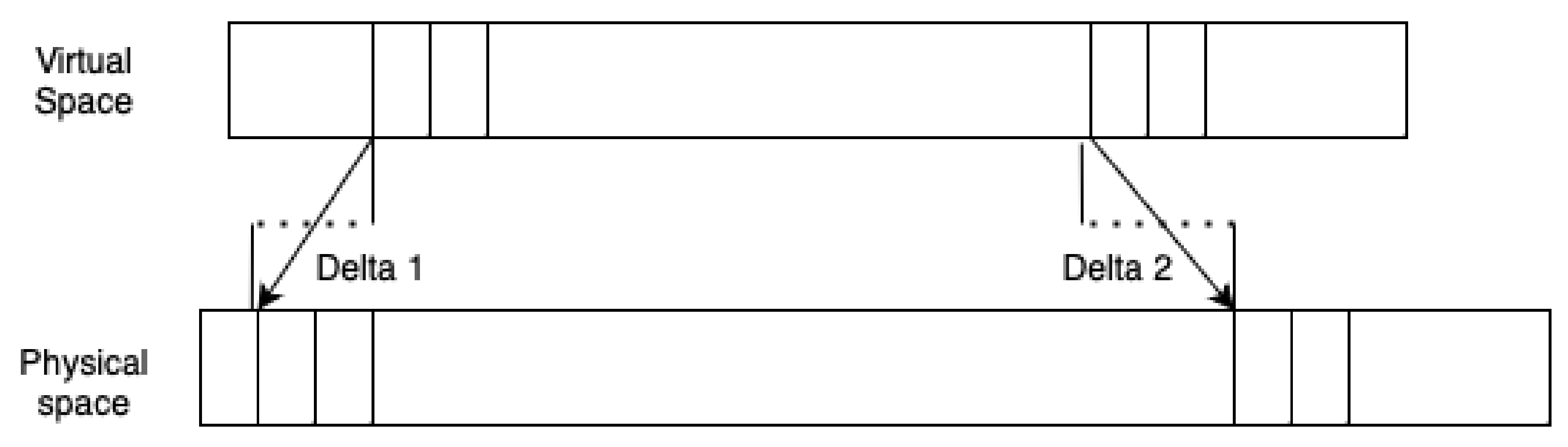
- (RQ1) Is there an optimization that explicitily uses FAT pointers to reduce the number of TLB misses to reduce the CPU STALL Time ?
- (RQ2) Can the number of L1 DTLB misses be minised by offloading translation lookups to the newly introduced range TLB ?

Expirement Proposal

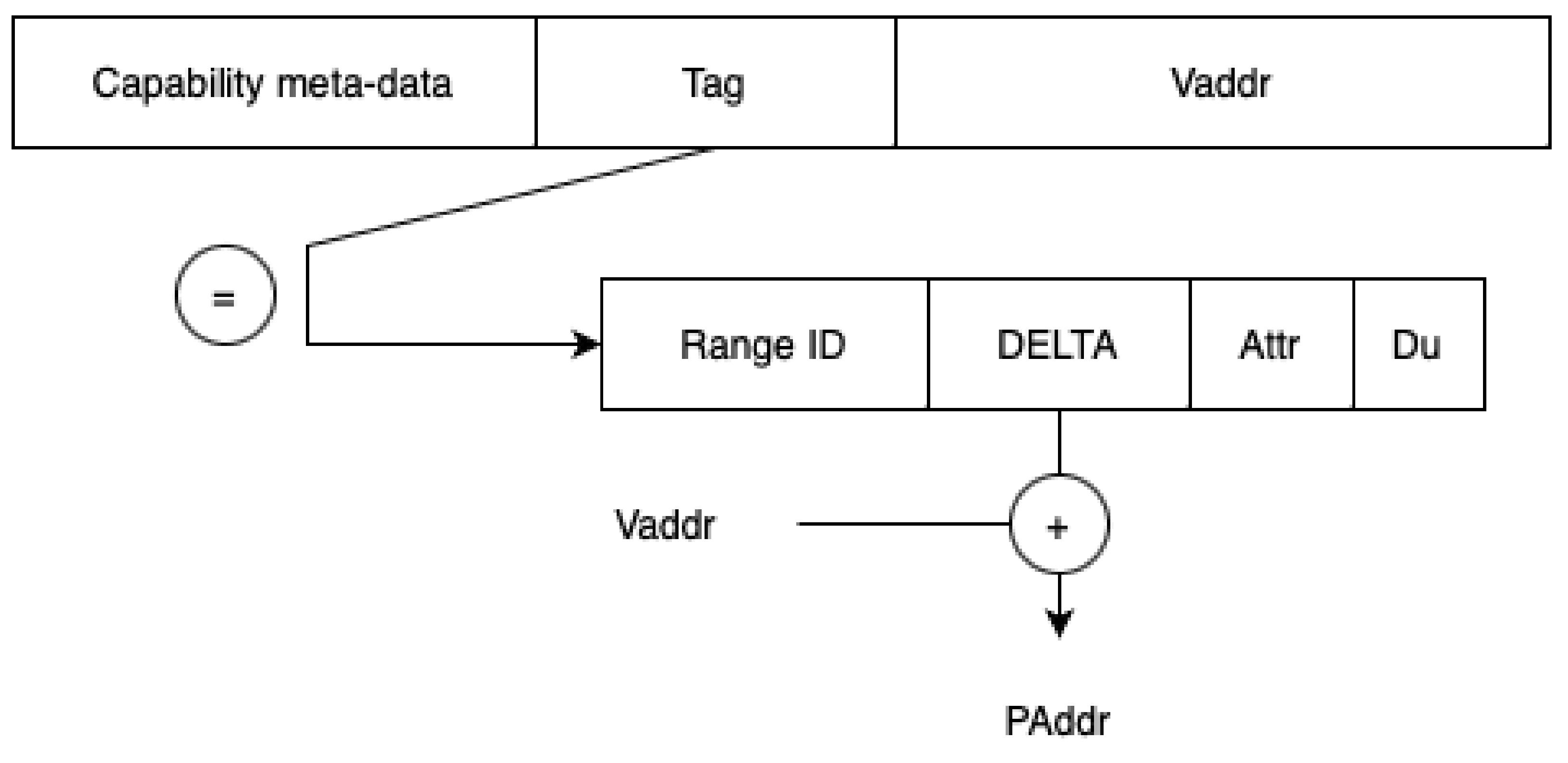

Different Cherified memory allocators.
2
Modified Malloc shared object file
loaded via LD to overwrite the existing implementation
1
C program (Pure Cap Binary)
Micro Benchmarks (The deciding factor would be C programs that aggressively use Malloc)
Links to the Modified MMAP call
3
Modified MMAP tested on different memory object sizes
24 kb
34 kb
48 kb
64 kb
....
The Range TLB implementation
allocation request
size ≦ threshold
malloc()
pointer tagging
tagged pointer
User
Kernel
size > threshold
mmap_range()
mmap()
Page Table
Range Table
Process Descriptor
1-bit validity tag
bounds compressed relative to address
63
0
perms
otype
bounds
X
RangeID: 13 bits
Address (46 bits)
Storing additional range IDs (Upto 8 bits available)

- *A set of Microbenchmarks which are designed to create pressure on the L1 TLB to create more L1 misses.*
- *Modified Malloc to call the new syscall mmap_range for large memory objects.*
- *Testing different memory object sizes to use range TLB.*

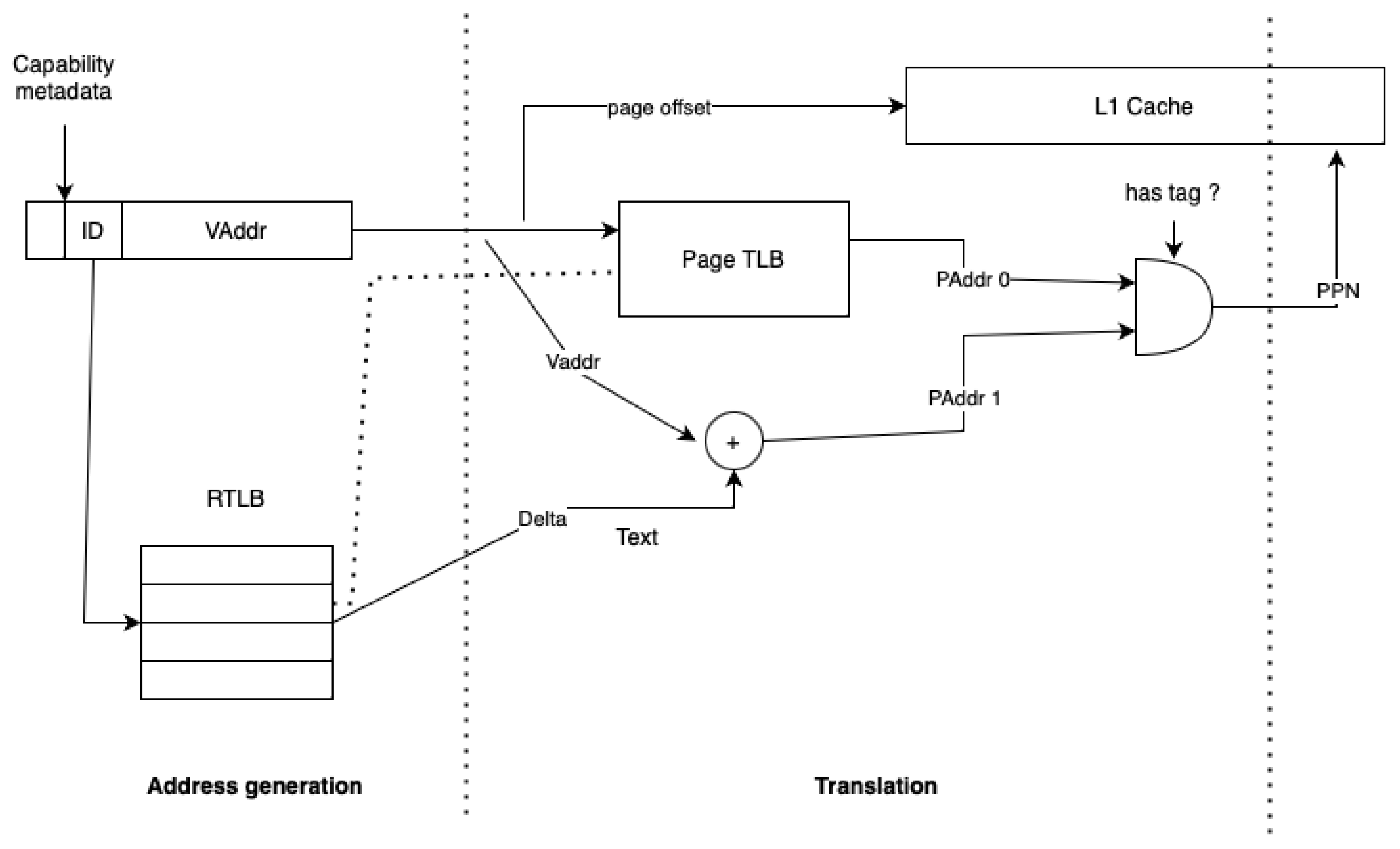
Range TLB



In the current expirement adopts the range translation concept from FlexPointer which was inturn adopted from RMM. A range consists of contigous virtual and phyiscal pages. Addresses share a common Delta (phyiscal address - virtual address). To assume physical contiguity of a particular range (We are proposing to use eager paging strategy as proposed from FlexPointer).



Range TLB is organised as a fully associative cache, As per the conventional the page table the lower 12 bits is used record attributes (eg: R/W: Read/Write). Du refers if the current range entry is a dummy one or not (This is used to indentify the next immediate space for a range entry).



Since the Range ID does not participate in the address generation. Using the ID provided the search for the delta value can occur on parallel to the address generation. Due to this the range TLB is shifted to a earlier stage.